MULTIMEDIA UNIVERSITY

STUDENT ID NO

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 2, 2016/2017

ECE3166 ADVANCED MICROPROCESSORS

(All Majors)

06 MARCH 2017
9:00A.M. – 11:00A.M.
(2 Hours)

INSTRUCTIONS TO STUDENTS

1. This Question paper consists of NINE pages with FOUR Questions only.
2. Attempt **ALL** questions. All questions carry equal marks and thedistribution of the marks for each question is given.
3. Please print all your answers in the Answer Booklet provided.

Question 1

(a) List THREE improved features of the Intel 80486 microprocessor compared to its predecessors.

[3 marks]

(b) State TWO advantages and TWO disadvantages of System-on-Chip (SoC).

[4 marks]

(c) Complete Table Q1 to contrast the data storage in a Little Endian microprocessor and a Big Endian microprocessor. Assume only one byte of data can be stored at each memory location. Store a double-word sized data for the following negative integer at address 3000H.

$$-123456789_{10}$$

Table Q1

Address	Little Endian	Big Endian
3000H		
3001H		
3002H		
3004H		

[4 marks]

(d) Propose the suitable ARM Cortex cores for the main processor, LTE modem and power management modules in a smartphone. Justify your proposal for each module.

[6 marks]

(e) Modern operating systems support multithreading, which is the ability to execute different parts of a program, simultaneously.

(i) Discuss TWO reasons of multithreading.

[3 marks]

(ii) Discuss how multithreading can be achieved in single-core processors.

[2 marks]

(iii) Multithreading can be implemented more effectively on multi-core processors. Describe TWO types of multi-core processor and explain their differences.

[3 marks]

Continued ...

Question 2

(a) Complete Table Q2 to state the addressing mode and physical address for the destination operands of the following x86 instructions.

Table Q2

Assembly instruction		Destination operand	
		Addressing mode	Physical address
(i)	`MOV [10H], AX`		
(ii)	`MOV 80H+[SI*2], AX`		
(iii)	`MOV ES:[BX][DI], AX`		

Initial contents of the registers:

`AX = A080H`	`CS = 3000H`
`BX = B020H`	`DS = 4000H`
`DI = 1000H`	`ES = 5000H`
`SI = 2000H`	`SS = 6000H`

[9 marks]

(b) Write a compact x86 assembly routine to compute the following equation. The final answer must be placed in register AX.

$$y = \sum_{n=1}^{50} 3(n+2)$$

[Hint: $y = 3(1+2)+3(2+2)+...+3(49+2)+3(50+2)$]

[8 marks]

(c) Develop an x86 assembly routine using repeat and string handling instructions to scan the memory from address ES:0300H to address ES: 03FFH for byte-sized data 5AH. When the first 5AH is discovered, exit from the loop and clear the carry flag.

[8 marks]

Continued ...

Question 3

(a) The Global Descriptor Table (GDT) is located at a linear address of 10003300H and capable of holding 2000 descriptors. Determine:

(i) The value loaded into Global Descriptor Table Register (GDTR).

[3 marks]

(ii) The address range of the last descriptor in GDT.

[4 marks]

(b) A code segment selector register is loaded with a value of 8106H. The base address of the Global Descriptor Table (GDT) and the Local Descriptor Table (LDT) are 0300 2200H and 0810 1000H, respectively.

[0300A300H] = 00FFH	[08109100H] = 03FFH
[0300A302H] = 0020H	[08109102H] = 0000H
[0300A304H] = 9210H	[08109104H] = DF26H
[0300A306H] = 0500H	[08109106H] = 0900H

(i) Find the Requested Privilege Level (RPL) and identify whether the descriptor is stored in the Global Descriptor Table (GDT) or Local Descriptor Table (LDT).

[3 marks]

(ii) Compute the address range of the code segment descriptor.

[3 marks]

(iii) Determine the base address and the limit of the segment.

[3 marks]

(iv) Examine and interpret the access right byte of the segment descriptor.

[4 marks]

(c) In protected mode, the IA-32 architecture provide privilege level checks for code and data access.

(i) If the Current Privilege Level (CPL) is 0, determine the accessible privilege level of data segment and justify your answer.

[4 marks]

(ii) Briefly explain the function of call gate.

[1 mark]

Continued ...

Question 4

(a) The Pentium processor is a superscalar processor produced by Intel.

(i) Briefly explain THREE major improvements of the Pentium microprocessor family.

[3 marks]

(ii) Discuss how MMX technology can speed up the performance of the Pentium processor.

[2 marks]

(b) Out-Of-Order execution (OOOE) is a technique that allows a superscalar processor to execute multiple independent instructions concurrently. Write-After-Read (WAR) and Write-After-Write (WAW) are two false dependencies in OOOE. Perform these tasks for the instruction sequence in Table Q4(b):

(i) Analyze the sequence of instructionsin Table Q4(b) to determine TWO pairs of instructions with WAR dependency and TWO pairs of instructions with WAW dependency.

[2 marks]

(ii) Register renaming is a technique to overcome these false dependencies. Discuss how does register renaming work.

[4 marks]

(iii) Show how register renaming can be used to solve WAR and WAW dependencies by populating Table Q4(b). Assume that the processor can execute up to FOUR instructions per cycle.

[6 marks]

Table Q4(b)

No.	Instruction	Cycle 1	Cycle 2
1	r1 ← mem1		
2	r2 ← r3 + r4		
3	r3 ← 7		
4	r4 ← mem2		
5	r3 ← r4 – r1		
6	r1 ← r2 + r4		

(c) Complete Table Q4(c) to show the affected register and its content after execution of the SSE instructions. Show the contents in **hexadecimal**.

[8 marks]

Table Q4(c)

Instruction	Affected register = Content (hexadecimal)
`MOVUPS XMM0, op_i`	`XMM0 = 40B00000 41200000 41780000 41A00000`
`MOVUPS XMM1, op_j`	`XMM1 = 41280000 41A40000 41F00000 42200000`
`MOVUPS XMM2, XMM1`	
`MOVUPS XMM3, XMM0`	
`MOVHLPS XMM2, XMM3`	
`MOVLHPS XMM3, XMM1`	

Continued...

Appendix A

Instruction Set Summary for Intel 8086/80186/80286/80386/80486

AAA - Ascii Adjust for Addition
AAS - Ascii Adjust for Subtraction
ADC - Add With Carry
ADD - Arithmetic Addition
AND - Logical And
BSF - Bit Scan Forward (386+)
BSR - Bit Scan Reverse (386+)
BSWAP - Byte Swap (486+)
BT - Bit Test (386+)
BTC - Bit Test with Compliment (386+)
BTR - Bit Test with Reset (386+)
BTS - Bit Test and Set (386+)
CALL - Procedure Call
CBW - Convert Byte to Word
CDQ - Convert Double to Quad (386+)
CLC - Clear Carry
CLD - Clear Direction Flag
CLI - Clear Interrupt Flag (disable)
CLTS - Clear Task Switched Flag (286+ privileged)
CMC - Complement Carry Flag
CMP - Compare
CMPS - Compare String (Byte, Word or Doubleword)
CMPXCHG - Compare and Exchange
CWD - Convert Word to Doubleword
CWDE - Convert Word to Extended Doubleword (386+)
DAA - Decimal Adjust for Addition
DAS - Decimal Adjust for Subtraction
DEC - Decrement
DIV - Divide
ENTER - Make Stack Frame (80188+)
ESC - Escape
HLT - Halt CPU
IDIV - Signed Integer Division
IMUL - Signed Multiply
IN - Input Byte or Word From Port
INC - Increment
INT - Interrupt
INTO - Interrupt on Overflow
INVD - Invalidate Cache (486+)
INVLPG - Invalidate Translation Look-Aside Buffer Entry (486+)
IRET/IRETD - Interrupt Return
Jxx - Jump Instructions Table
JCXZ/JECXZ - Jump if Register (E)CX is Zero
JMP - Unconditional Jump
LAHF - Load Register AH From Flags
LAR - Load Access Rights (286+ protected)
LDS - Load Pointer Using DS
LEA - Load Effective Address
LES - Load Pointer Using ES
LFS - Load Pointer Using FS (386+)
LGDT - Load Global Descriptor Table (286+ privileged)
LIDT - Load Interrupt Descriptor Table (286+ privileged)
LGS - Load Pointer Using GS (386+)
LLDT - Load Local Descriptor Table (286+ privileged)
LMSW - Load Machine Status Word (286+ privileged)

Continued ...

LOCK - Lock Bus
LODS - Load String (Byte, Word or Double)
LOOP - Decrement CX and Loop if CX Not Zero
LOOPE/LOOPZ - Loop While Equal / Loop While Zero
LOOPNZ/LOOPNE - Loop While Not Zero / Loop While Not Equal
LSL - Load Segment Limit (286+ protected)
LSS - Load Pointer Using SS (386+)
LTR - Load Task Register (286+ privileged)
MOV - Move Byte or Word
MOVS - Move String (Byte or Word)
MOVSX - Move with Sign Extend (386+)
MOVZX - Move with Zero Extend (386+)
MUL - Unsigned Multiply
NEG - Two's Complement Negation
NOP - No Operation (90h)
OR - Inclusive Logical OR
OUT - Output Data to Port
POP - Pop Word off Stack
POPA/POPAD - Pop All Registers onto Stack (80188+)
POPF/POPFD - Pop Flags off Stack
PUSH - Push Word onto Stack
PUSHA/PUSHAD - Push All Registers onto Stack (80188+)
PUSHF/PUSHFD - Push Flags onto Stack
RCL - Rotate Through Carry Left
RCR - Rotate Through Carry Right
REP - Repeat String Operation
REPE/REPZ - Repeat Equal / Repeat Zero
REPNE/REPNZ - Repeat Not Equal / Repeat Not Zero
RET/RETF - Return From Procedure
ROL - Rotate Left
ROR - Rotate Right
SAHF - Store AH Register into FLAGS
SAL/SHL - Shift Arithmetic Left / Shift Logical Left
SAR - Shift Arithmetic Right
SBB - Subtract with Borrow/Carry
SCAS - Scan String (Byte, Word or Doubleword)
SETS - Set if Signed (386+)
SETNS - Set if Not Signed (386+)
SETC - Set if Carry (386+)
SETNC - Set if Not Carry (386+)
SETO - Set if Overflow (386+)
SETP/SETPE - Set if Parity / Set if Parity Even (386+)
SETNP/SETPO - Set if No Parity / Set if Parity Odd (386+)
SGDT - Store Global Descriptor Table (286+ privileged)
SIDT - Store Interrupt Descriptor Table (286+ privileged)
SHL - Shift Logical Left
SHR - Shift Logical Right
SHLD/SHRD - Double Precision Shift (386+)
SLDT - Store Local Descriptor Table (286+ privileged)
SMSW - Store Machine Status Word (286+ privileged)
STC - Set Carry
STD - Set Direction Flag
STI - Set Interrupt Flag (Enable Interrupts)
STOS - Store String (Byte, Word or Doubleword)
STR - Store Task Register (286+ privileged)
SUB - Subtract
TEST - Test for Bit Pattern
WBINVD - Write-Back and Invalidate Cache (486+)
XCHG - Exchange
XOR - Exclusive OR

Continued ...

Mnemonic	Meaning	Format	Operation	Flags Affected
MOVS	Move string	MOVSB/MOVSW	((ES)0 + (DI)) ← ((DS)0 + (SI)) (SI) ← (SI) ± 1 or 2 (DI) ← (DI) ± 1 or 2	None
CMPS	Compare string	CMPSB/CMPSW	Set flags as per ((DS)0 + (SI)) − ((ES)0 + (DI)) (SI) ← (SI) ± 1 or 2 (DI) ← (DI) ± 1 or 2	CF, PF, AF, ZF, SF, OF
SCAS	Scan string	SCASB/SCASW	Set flags as per (AL or AX) − ((ES)0 + (DI)) (DI) ← (DI) ± 1 or 2	CF, PF, AF, ZF, SF, OF
LODS	Load string	LODSB/LODSW	(AL or AX) ← ((DS)0 + (SI)) (SI) ← (SI) ± 1 or 2	None
STOS	Store string	STOSB/STOSW	((ES)0 + (DI)) ← (AL or AX) ± 1 or 2 (DI) ← (DI) ± 1 or 2	None

Prefix	Used with:	Meaning
REP	MOVS STOS	Repeat while not end of string CX ≠ 0
REPE/REPZ	CMPS SCAS	Repeat while not end of string and strings are equal CX ≠ 0 and ZF = 1
REPNE/REPNZ	CMPS SCAS	Repeat while not end of string and strings are not equal CX ≠ 0 and ZF = 0

Appendix B

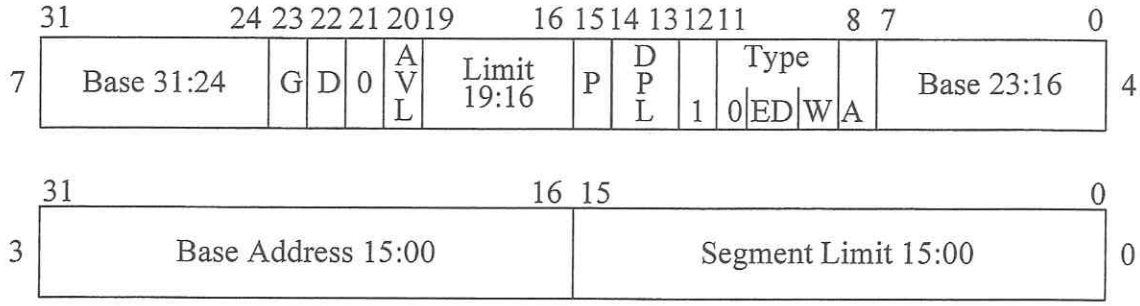
Summary of Selected SSE Instructions

Mnemonic	Description
MOVAPS	Move aligned packed single-precision floating-point values
MOVUPS	Move unaligned packed single-precision, floating-point
MOVHPS	Move high packed single-precision floating-point
MOVHLPS	Move packed single-precision floating-point high to low
MOVLHPS	Move packed single-precision floating-point low to high
MOVLPS	Move low packed single-precision floating-point
MOVMSKPS	Move packed single-precision floating-point mask
MOVSS	Move scalar single-precision floating-point
ADDSS	Scalar add pair of operands
ADDPS	Packed add pairs of operands

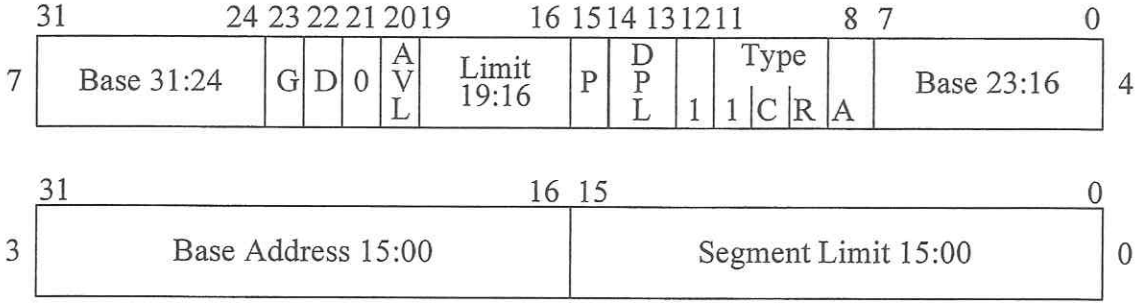
Continued...

Appendix C

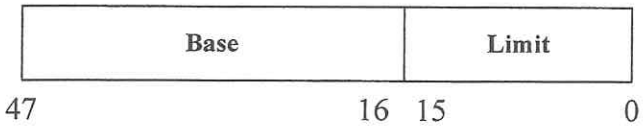
Data-Segment Descriptor:



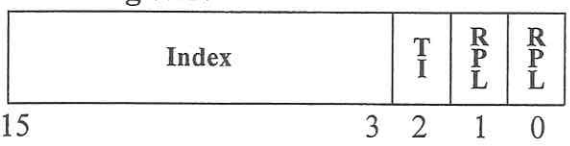
Code-Segment Descriptor:



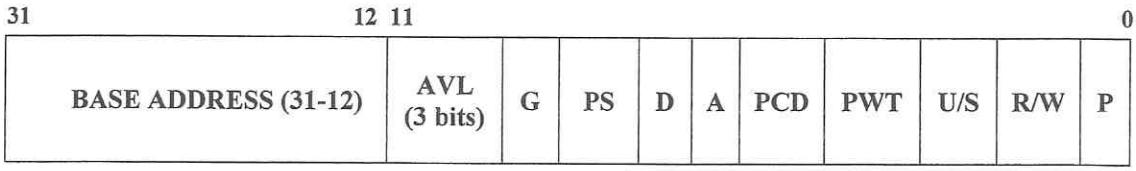
Global Descriptor Table Register (GDTR) and Interrupt Descriptor Table Register (IDTR):



Selector Register:



Page Directory or Page Table Entry Format:



End of Paper